IN THE SPECIFICATION:

Please replace the FIRST full paragraph of specification page PAGE 10 with the following replacement paragraph:

— Fig. 3 shows an exemplary data structure 300 after a file data block has been modified. In this illustrative example, file data block 120C was modified to file data block 120C'. When file data block 120C is modified file data block 120C', the contents of the modified file data block are written to a new location on disk as a function for the exemplary file system. Because of this new location, the inode file data block 315 pointing to the revised file data block 120C must be modified to reflect the new location of the file data block 120C. Similarly, the inode file indirect block 310 must be rewritten to point to the newly revised inode file and data block. Thus, after a file data block has been modified the PCPI inode 205 contains a pointer to the original inode file system indirect block 110 which in turn contains a link to the inode file data block 115. This inode file data block 115 contains pointers to the original file data blocks 120A, 120B and 120C. However, the newly written inode file data block 315 includes pointers to unmodified file data blocks 120A and 120B. The inode file data block 315 also contains a pointer to the modified file data block 120C' representing the new arrangement of the active file system. A new file system root inode 305 is established representing the new structure 300. Note that metadata (not shown) stored in any Snapshotted blocks (e.g., 205, 110, and 120C) protects these blocks from being recycled or overwritten until they are released from all PCPIs. Thus, while the active file system root inode 305 points to new blocks 310, 315 and 120C', the old blocks 205, 110, 115 and 120C are retained until the PCPI is fully released. —

Please replace the paragraph beginning on line 1 of page 12 with the following replacement paragraph.

— In brief summary, the source creates a pair of discrete time-separated PCPIs of the volume. These can be created as part of the commit process in which data is committed to non-volatile memory in the filer or by another mechanism. The "new" PCPI 410 is a recent PCPI of the volume's active file system. The "old" PCPI 412 is an older PCPI of the volume, which should match the image of the file system mirrored/replicated on the destination mirror. Note that the file server is free to continue work on new file service requests once the new PCPI 412 is made. The new PCPI acts as a checkpoint of activity up to that time rather than an absolute representation of the then-current volume state. A differencer 420 scans the blocks 422 in the old and new PCPIs. In particular, the differencer works in a block-by-block fashion, examining the list of blocks in each PCPI to compare which blocks have been allocated. In the case of a write-anywhere system, the block is not reused as long as a PCPI references it, thus a change in data is written to a new block. Where a change is identified (denoted by a presence or absence of an 'X' designating data), a decision process ~~400~~500, shown in Fig. 5, in the differencer 420 decides whether to transmit the data to the destination 402. The decision process 500 compares the old and new blocks as follows: (a) where data is in neither an old nor new block (case 502) as in old/new block pair 430, no data is available to transfer; (b) where data is in the old block, but not the new (case 504) as in old/new block pair 432, such data has already been transferred, (and any new destination PCPI pointers will ignore it), so the new block state is not transmitted; (c) where data is present in the both the old block and the new block (case 506) as in the old/new block pair 434, no change has occurred and the block data has already been transferred in a previous PCPI; and (d) finally, where the data is not in the old block, but is in the new block (case 508) as in old/new block pair 436, then a changed data block is transferred over the network to become part of the changed volume mirror/replica set 440 at the destination as a changed block 442. In the exemplary write-anywhere arrangement, the changed blocks are written to new, unused locations in the storage array. Once all changed blocks are written, a base file system information block, that is the root pointer of the new PCPI, is then committed to the destination. The transmitted file system information block is committed, and updates the overall destination file system by pointing to the changed block structure in the destination, and replacing the previous file system information block. The changes are at this point committed as the latest incremental update of the destination volume mirror. This file system accurately represents the "new" mirror on the source. In time a new "new" mirror is created from further incremental changes. —

Please replace the paragraph beginning on page PAGE 17, LINE 24 of the speciation with the following replacement paragraph:

— To facilitate access to the disks, the storage operating system 800 implements a write-anywhere file system that logically organizes the information as a hierarchical structure of directory, file and vdisk objects (hereinafter "directories", "files" and "vdisks") on the disks. A vdisk is a special file type that is translated into an emulated disk or logical unit number (lun) as viewed by a storage area network (SAN) client. Each "on-disk" file may be implemented as set of disk blocks configured to store information, such as data, whereas the directory may be implemented as a specially formatted file in which names and links to other files and directories are stored. Vdisks are further described in U.S. Patent Application Serial No. 10/216,453, entitled STORAGE VIRTUALIZATION BY LAYERING VIRTUAL DISK OBJECTS ON A FILE SYSTEM, by Vijayan Rajan, et al. the teachings of which are hereby incorporated by reference. —